Received: 7 January 2022 | Revised: 23 February 2022 | Accepted: 9 March 2022

DOI: 10.1002/mma.8254

RESEARCH ARTICLE

WILEY

Granular sieving algorithm for selecting best n parameters

Tao Qian[1] | **Lei Dai**[2] | **Liming Zhang**[2] | **Zehua Chen**[3]

[1]Macao Center for Mathematical Sciences, Macau University of Science and Technology, Taipa, Macao

[2]Faculty of Science and Technology, University of Macau, Taipa, Macao

[3]College of Data Science, Taiyuan University of Technology, Taiyuan, China

Correspondence
Liming Zhang, Faculty of Science and Technology, University of Macau, Taipa, Macao.
Email: lmzhang@um.edu.mo

Communicated by: W. Sprößig

Funding information
Multi-year Research Grant, Grant/Award Number: MYRG2018-00111-FST; Science and Technology Development Fund of Macau SAR, Grant/Award Number: 0060/2021/A and 0123/2018/A3

A common type problem of optimization is to find simultaneously n parameters that globally minimize an objective function of n variables. Such problems are seen in signal and image processing and in various applications of mathematical analysis of several complex variables and Clifford algebras. Objective functions are usually assumed to be Lipschitzian with maybe unknown Lipschitz constants. A number of methods have been established to discard the sets called "bad sets" in a partition that is impossible to contain any optimal point, as well as to treat the unknown Lipschitz bound problem along with the algorithm. In the present paper, a simple criterion of eliminating bad sets is proposed for the first time. The elimination method leads to a concise and rigorous proof of convergence. The algorithm, on the range space side, converges to the global minimum with an exponential rate, while on the domain space side, converges with equal accuracy to the set of all the global minimizers. To treat the unknown Lipschitz constant dilemma, we propose a practical pseudo-Lipshitz bound process. The methodology is of fundamental nature with straightforward mathematical formulation applicable to multivariate objective functions defined on any compactly connected manifolds in higher dimensions. The method is tested against an extensive number of benchmark functions in the literature. The experimental results exhibit considerable effectiveness and applicability of the algorithm.

KEYWORDS

deterministic method in global optimization, global minimum and minimizer, Lipschitz condition, partition of set

MSC CLASSIFICATION

90C26; 46N10; 26A16

1 | INTRODUCTION

The objective functions that we study in this work are of the form $f(x)$, where f is a real-valued Lipschitz continuous function with the variables $x = (x_1, \cdots, x_n)$ ranging in a connected compact (bounded and closed) set C of a real- or complex- n-dimensional Euclidean space. Precisely, the function is required to satisfy the Lipschitz condition: There exists

The first author invented the theory. The second author, being considered to give equal contribution to the paper, realized the algorithm and conducted the enormous experimental examples.

a constant L such that

$$\frac{|f(x+\Delta x)-f(x)|}{|\Delta x|} \le L. \tag{1}$$

We call the difference quotient quantity on the left-hand side of (1) as *relative variation*, and L a Lipschitz *bound* of f. The infimum of the Lipschitz bounds L is denoted by L_0 and called the *Lipschitz constant of* f. There hold the relations

$$\begin{aligned} L_0 &= \min\{L \ : \ L \text{ is an upper bound of the relative variations}\} \\ &= \sup\left\{ \frac{|f(x+\Delta x)-f(x)|}{|\Delta x|} \ : \ x, x+\Delta x \in C, \Delta x \neq 0 \right\}. \end{aligned}$$

A point x^* is said to be a *global minimizer* of the function $f(x)$ defined on C, if $x^* \in C$ and $f(x^*) \le f(x)$ for all $x \in C$. In such case, $f(x^*)$ is called the *global minimum* of f on C and denoted as $f_{\min}$. In below we denote $v = f_{min}$. Lipschitzian functions are surely continuous but may not be differentiable. In this study, in addition to satisfying the Lipschitz condition, no other global information about the problem (such as Lipschitz constant, gradient, and convexity) is needed.

The proposed algorithm is inspired partially by our continuing study on the open problem of finding best n kernel approximation in certain reproducing kernel Hilbert spaces,[1–3] and partially by the granular computing idea in artificial intelligence.[4–6] It is thus named as granular sieving (GrS) algorithm. It is seen that the type of optimization problems for objective functions defined on open or even unbounded open sets can possibly adopt our GrS algorithm or some existing Lipschitz global optimization algorithms if the boundary values are well controlled.

The related study on Lipschitz bounds shows that when L_1 is less than the Lipschitz constant, L_0, accuracy of the solution increases as L_1 approaches the Lipschitz constant from below. On the other hand, when L_1 is greater than the Lipschitz constant, the computational efficiency increases as L_1 approaches the Lipschitz constant from above. In applications, one seeks for Lipschitz bounds L that are close to their infimum value L_0 (close Lipschitz bounds). Seeking for close Lipschitz bounds can be independent research, it is, however, often associated with the algorithm in use.[7–9] To realize the GrS algorithm with unknown Lipschitz constant one can adopts the existing methods for estimating close Lipschitz bounds, which is mostly adequate for the algorithm. This includes, for instance, at the beginning of the algorithm, estimating close Lipschitz bounds in the whole domain.[10,11] In the literature, it is more common and more effective if one adaptively re-estimates it during the search at each iteration.[12–14] As examples, the DIRECT method[15] and the GOSH[7] method use, at each iteration, several estimates of the Lipschitz constant simultaneously. The proposed GrS can also do the re-estimating along with its iterative progress. On top of the commonly used methods, in this study, we incorporate what we called *pseudo-Lipschitz process*. The particular feature of the method is: Along with finding the optimization solution, it judges whether a constant L_1 is a suitable replacement of L_0 in the algorithm. One also keeps in mind that in the differentiable case the global minimum normally occurs at points with locally small Lipschitz bounds.

Here is a short description of the pseudo-Lipschitz bound process as part of our algorithm when the Lipschitz is unknown. The proposed process is based on the following result (see Theorem 2): Any positive number L_1 one attempts to replace the Lipschitz constant L_0 in the algorithm guaranteed by Theorem 1 can lead to a solution. The algorithm, as a matter of fact, does not require L_1 to be a true bound of the relative variations of the objective function. It is shown that for any $L_1 > 0$ the GrS algorithm converges; and if $L_1 < L_2$, then $v^{L_1}(f) \ge v^{L_2}(f)$, where $v^{L_1}(f)$ denotes the resulted pseudo-global minimum of f by using L_1 as a replacement of L_0 (*a pseudo-Lipschitz bound*) in the algorithm. A constant L_1 that gives rise to the right global minimum is said to be a *suitable pseudo-Lipschitz bound*. Based on these results the stability given by $v^{L_1}(f) = v^{L_2}(f)$ is a necessary condition, and thus an indication of suitability of L_1. To summarize, besides the theoretical brevity and clarity, uniformness for dimensions and shapes of domains, and uniform fast convergence to the global minimum, GrS is incorporated with a pseudo-process to solve the unknown Lipschitz constant black-box problem.

In the application aspect, the authors were motivated by the classical problem of best rational approximation to functions in the Hardy space of the unit disc $\mathbf{D}$ by using rational functions of degrees not exceeding n, and generalizations to various types of analytical reproducing kernel Hilbert spaces. In particular, this topic recently has been extended to Clifford algebra. n-best parameters selection has deep and wide connections to the current learning theory used to signal and image processing, to system identification, as well as to numerical solutions of ordinary and partial differential solutions. In many cases, including the weighted Bergman and the weighted Hardy spaces, existence of n-kernel best approximation has been proved. In contrast, however, an algorithm to ultimately find a practical best set of n parameters even in the Hardy space case has yet been remained as an open problem. The analytical methods recently developed in Wang and

Qian[16] now can help to reduce the n best rational problem originally posted in the open unit disc $\mathbf{D}$ to a compact disc inside $\mathbf{D}$ and thus to solve the algorithm problem by means of the introduced GrS method. This will be our forthcoming work.

The paper is structured as follows. In Section 2, the GrS algorithm is introduced and the convergence of the algorithm is proved. Multiple estimation process in relation to pseudo-Lipschitz bounds is presented in Section 3. Numerical experiments and results analysis are described in Sections 4 and 5. Conclusions are drawn in Section 6.

2 | LITERATURE REVIEW

A great amount of literature have devoted to the Lipschitzian global optimization problems.[7,9,11,15,17–28]

The common idea is to use different partition strategies to eliminate, based on certain criteria, the "bad" sets in each of a sequence carefully designed partitions of the domain, or the corresponding remaining parts of the domain, in which there is no possibility to have global minimizers. In the eliminating process what are left, the "good" sets, are for further analysis to extract out the minimizers.

We recall that Lera and Sergeyev recently carried out an interesting study of Lipschitz global optimization that is based on the abstract and classical mathematical object Peano space-filling curves.[7] The study effectively reduces the higher dimensional problems to the one-dimensional ones, but with the cost of reduction of the Lipschitz continuity (with unknown Lipschitz constant) to one of the Hölder type continuities (with unknown Hölder constant as well), which is caused from the fractal nature of the space-filling curves. It is noted that the Hölder continuity does not imply boundedness of gradients when differentiable.

In particular, Al-Dujaili et al. under their bound-constrained black-box global optimization (BCBBGOP) formulation proposed MSO (Multi-scale optimization) scheme dividing the similar problems into two groups of which one depends on precise information of smoothness of the objective function, including for example, LO, DOO, and the other does not require precise information of the smoothness, including DIRECT, MCS, SOO, etc. In the BCBBGOP dividing, the proposed GrS algorithm falls into the second category. The GrS algorithm, however, has the character that the bad and good sets are justified by a single and yet simple inequality criterion and, as a result, leads to a half-page concise and rigorous mathematical proof for the convergence, being applicable to any connected compact set as domain of the objective function. The proof in particular indicates that the kth level good sets converge to the entire solution set, and the evaluations of the points in the kth level good sets are all within an error, to the global minimum, less than $C\delta^k$ for some $\delta \in (0, 1/2]$.

3 | THE GRS ALGORITHM

We will be working with a real-valued continuous objective function with real variables. For complex variables the theory and algorithm are similar. Let C, being the domain of the objective function under study, be a compact (closed and bounded) path-wise connected set in the Euclidean space $\mathbb{R}^n$. The algorithm will construct a decreasing sequence of compact sets $C = C_1 \supset \cdots \supset C_k \supset \cdots$ in the domain C of the objective function with the following properties. For each $k = 1, \cdots$, one constructs a partition of C_k, $\mathbb{N}_k = \{C_l^{(k)} : l = 1, \cdots, l_k\}$, such that $C_k = \cup_{l=1}^{l_k} C_l^{(k)}$. For each fixed k the pairs $(k, l), l = 1, \cdots, l_k$, and no others, are called *k-admissible sets*, or just *admissible sets*, in brief. The sets $C_l^{(k)}$ satisfy the following conditions:

(i) When k grows larger, $\mathbb{N}_k$ becomes finer in the sense that a set $C_{l'}^{(k+1)}$ is either entirely contained in $C_l^{(k)}$ or has empty overlap with the interior of $C_l^{(k)}$.
(ii) The maximal diameter of $C_l^{(k)}, l = 1, \cdots, l_k$, denoted as δ_k, tends to zero along with $k \to \infty$. The diameter is defined as the maximal distance between any two points in the set.
(iii) Each set $C_l^{(k)}$ is compact whose boundary can have non-empty overlap with the boundaries of the other $C_{l'}^{(k')}$.
(iv) Along with the construction we name in each $C_l^{(k)}$ a representative point $x_l^{(k)}$. When $C_l^{(k)}$ is geometrically symmetric we may select $x_l^{(k)}$ as the geometrical center of $C_l^{(k)}$. For this reason we, in below, call $x_l^{(k)}$ as "center" of $C_l^{(k)}$, although they are not necessarily the centers, but only representative points of $C_l^{(k)}$.

We will be using mappings between sets defined as

$$f(A) = \{b \in \mathbf{R} : \exists a \in A, b = f(a)\},\ A \subset C,$$

where **R** stands for the set of real numbers.

As a fundamental result of a continuous function defined in a compact set, there exists a non-empty compact set $\tilde{C}$ on which the function takes the global minimum v, that is $f(\tilde{C}) = \{v\}$. The purpose of the algorithm is to inductively construct the set C_{k+1} through some sets in the partition $\mathbb{N}_k = \{C_l^{(k)} : l = 1, \cdots, l_k\}$ of C_k, and to realize $\tilde{C} = \cap_{k=1}^{\infty} C_k$, and $f(\tilde{C}) = \{v\}$. The proposed GrS algorithm for finding the global minimum v and at the same time all the global minimizers is given below: For each fixed level k the set C_k is expressible by the sets in its partition: $C_k = \cup_{l=1}^{l_k} C_l^{(k)}$, $C_l^{(k)} \in \mathbb{N}_k, l = 1, \cdots, l_k$. For the fixed k, exam all the function values at the center points $x_l^{(k)}$ of the k-level sets $C_l^{(k)}$, and select the minimum one among the function values $f(x_l^{(k)})$ denoted by v_k. If a center $x_{l'}^{(k)}$ satisfies the relation $f(x_{l'}^{(k)}) > v_k$, we can determine by using the Lipschitz condition whether there would be some function values $f(x), x \in C_{l'}^{(k)}$, that can possibly be equal to, or below v_k. The k-level sets $C_{l'}^{(k)}$ that are of such possibility will be called "good sets." If there are no function values that could be equal to or below the value v_k, then we simply delete the whole set $C_{l'}^{(k)}$. The deleted sets $C_{l'}^{(k)}$ are referred as "bad sets." The union of the "good sets" is defined to be our next generation set C_{k+1}. The algorithm consists of iteration of the above sorting process that forms C_{k+1} as union of the good subsets in $\mathbb{N}_k$ of the partition of C_k. Figure 1 shows, as an example, the first three consecutive partitions of a two-dimensional function. Figure 1A shows the first partition, and Figure 1B,C illustrates that only those classified as "good sets" are collected and further partitioned, expressed as the light-blue boxes in Figure 1B and the dark-blue boxes in Figure 1C. Finally one takes $\cap_{k=1}^{\infty} C_k$, which is non-empty giving rise to the global minimum v. Below is a more detailed explanation of the algorithm.

Carrying out the above mentioned sorting at the level $k = 1$ is based on constructing a partition $\mathbb{N}_1$ of $C = C_1$. The partition should be fine enough so that the maximal diameter δ_1 of all $C_l^{(1)}$ makes the quantity $\delta_1 L_0$ strictly smaller than the largest possible variation of the function values $f(x_l^{(1)})$ from their minimum value v_1, where

$$v_1 = \min\{f(x_l^{(1)}) : l = 1, \cdots, l_1\}. \tag{2}$$

We divide the sets in the C_1-partition $\mathbb{N}_1$ into two non-overlap subclasses: $\mathbb{N}_1 = \mathbb{B}_1 \cup \mathbb{G}_1$.

The subclass $\mathbb{B}_1$, as the collection of the "bad" sets, consists of the sets $C_l^{(1)}$ whose centers $x_l^{(1)}$ satisfy

$$f(x_l^{(1)}) > v_1 + \delta_1 L_0. \tag{3}$$

Notice that for any $C_l^{(1)}$ the quantity $\delta_1 L_0$ is the greatest possible variation of the function values in the set $C_l^{(1)}$ from its center. The inequality (3) sorts out those $C_l^{(1)}$ in which any function value cannot reach the minimum level v_1, let alone below it. The corresponding set $C_l^{(1)}$ then should be deleted. If no sets are classified into $\mathbb{B}_1$, it means that the partition $\mathbb{N}_1$ is not fine enough, or δ_1 is not fine enough.

The subclass $\mathbb{G}_1$, as the collection of the "good" sets, consists of those $C_l^{(1)}$ whose centers $x_l^{(1)}$ satisfy the opposite inequality

$$f(x_l^{(1)}) \leq v_1 + \delta_1 L_0. \tag{4}$$

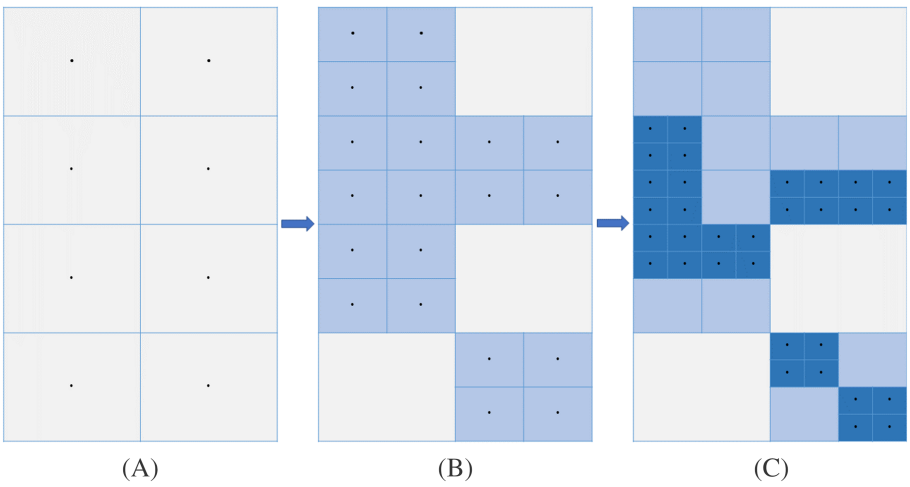


FIGURE 1 An example of the first three consecutive partition processes of a two-dimensional function. The black dots represent the centers of the boxes [Colour figure can be viewed at wileyonlinelibrary.com]

If a set $C_l^{(1)}$ belongs to the collection $\mathbb{G}_1$, due to the inequality (4), one cannot exclude the possibility that some function values of $x \in C_l^{(1)}$ can possibly reach or even below the minimum v_k. Those sets $C_l^{(1)}$ should be kept and further analyzed.

We eliminate all the sets in the subclass $\mathbb{B}_1$ of the bad sets and union those in the subclass $\mathbb{G}_1$ of the good sets. Define $C_2 = \cup\{C_l^{(1)} \ : \ C_l^{(1)} \in \mathbb{G}_1\}$, and construct a partition $\mathbb{N}_2$ of C_2 finer than what is inherited from $\mathbb{N}_1$.

With a suitable partition $\mathbb{N}_2$, $C_2 = \cup_{l=1}^{l_2} C_l^{(2)}$. Let δ_2 be the maximal diameter of the subsets in the partition $\mathbb{N}_2$. We define

$$v_2 = \min\{f(x_l^{(2)}) \ : \ l = 1, \cdots, l_2\}. \tag{5}$$

Using the criteria

$$f(x_l^{(2)}) > v_2 + \delta_2 L_0 \tag{6}$$

and

$$f(x_l^{(2)}) \le v_2 + \delta_2 L_0 \tag{7}$$

we divide the sets $C_l^{(2)}$ in the partition $\mathbb{N}_2$ into a bad set subclass $\mathbb{B}_2$ and a good set subclass $\mathbb{G}_2$, respectively. Eliminate all the bad sets $C_l^{(2)}$ whose centers satisfying the inequality (6) and keep all the good sets as the collection $\mathbb{G}_2$. Define $C_3 = \cup\{C_l^{(2)} \ : \ C_l^{(2)} \in \mathbb{G}_2\}$ and construct $\mathbb{N}_3$ as a finer partition of C_3, and so on. Figure 2 shows the granular sieving process of a one-dimensional function at the kth and$(k+1)$th partition levels.

Repeating this process inductively with $C_{k+1} = \cup_{C_l^{(k)} \in \mathbb{G}_k} C_l^{(k)}$ and making use the condition $\delta_k \to 0$, one has

Theorem 1. *$\cap_{k=1}^{\infty} C_k = \tilde{C} \neq \emptyset$, and $f(\tilde{C}) = \{v\}$.*

Proof of Theorem 1. We note that for each k, the set $C_l^{(k)} \in \mathbb{G}_k$ if and only if

$$f(x_l^{(k)}) \le v_k + \delta_k L_0. \tag{8}$$

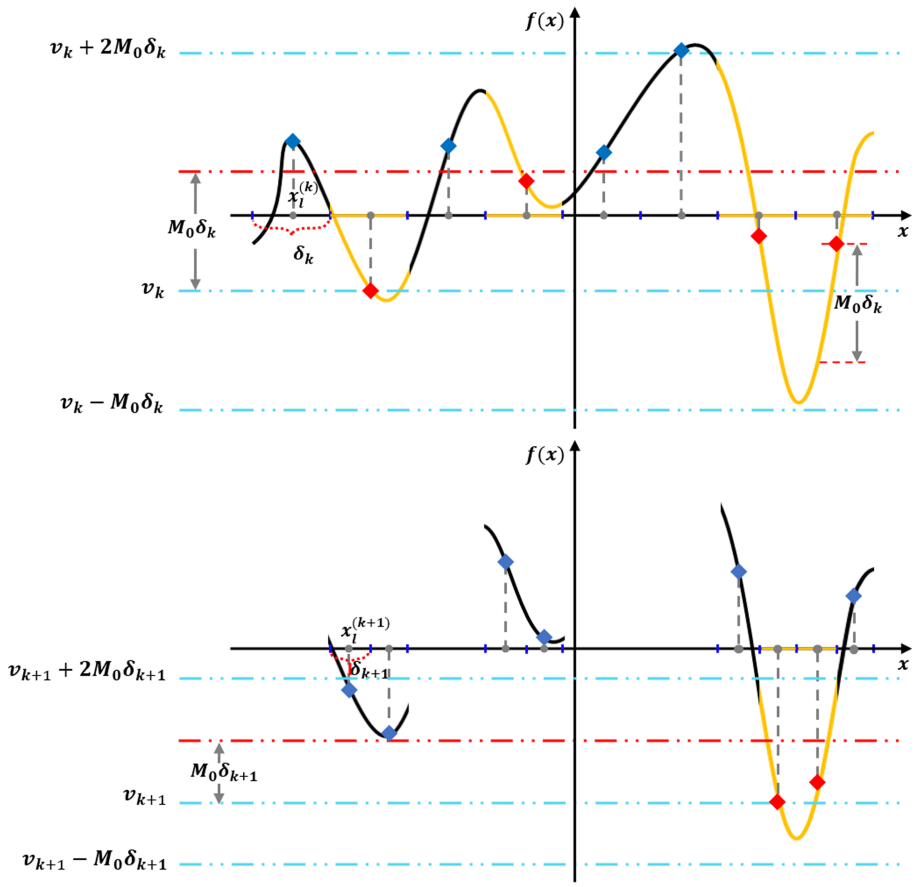


FIGURE 2 Demonstration diagram of the granular sieving process of a one-dimensional function at the kth and $(k+1)$th partition levels. On the x axis, the gray dots are centers of the granule sets in the partition sets. The yellow and black intervals and function graphs represent, respectively, the "good" and "bad" sets in the domain and the corresponding function values. The red and blue diamonds represent the function values of the partition centers of the "good" and "bad" sets, respectively. The "good sets" are those with partition centers below the red dotted line, while the "bad sets" are those with partition centers above the red dotted line [Colour figure can be viewed at wileyonlinelibrary.com]

We also have,

$$|f(x) - f(x_l^{(k)})| \leq \delta_k L_0. \tag{9}$$

They imply

$$f(C_l^{(k)}) \subset [v_k - \delta_k L_0, v_k + 2\delta_k L_0] = I_k^{L_0}.$$

Therefore,

$$f(C_{k+1}) = f(\cup_{C_l^{(k)} \in \mathbb{G}_k} C_l^{(k)}) = \cup_{l=1}^{l_k} f(C_l^{(k)}) \subset I_k^{L_0}.$$

There holds

$$f(\cap_{k=1}^{\infty} C_{k+1}) = \cap_{k=1}^{\infty} f(C_{k+1}) \subset \cap_{k=1}^{\infty} I_k^{L_0}.$$

Since the compact sets sequence $C_2 \supset C_3 \supset \cdots C_{k+1} \supset \cdots$ has the finite-non-empty-intersection property, $C \supset \tilde{C} = \cap_{k=1}^{\infty} C_{k+1} \neq \emptyset$.[29] On the other hand, $v \in I_k^{L_0}$ for each k, and $|I_k^{L_0}| = 3\delta_k L_0 \to 0$, we have $\{v\} = \cap_{k=1}^{\infty} I_k^{L_0}$. Therefore,

$$f(\tilde{C}) = \{v\}.$$

The proof is complete. □

4 | MULTIPLE ESTIMATION PROCESS UNDER PSEUDO-LIPSCHITZ BOUNDS

The above process requires the partitions $\mathbb{N}_1, \mathbb{N}_2, \cdots, \mathbb{N}_k, \cdots$ to satisfy that for each k both the inequalities

$$f(x_l^{(k)}) - v_k \leq \delta_k L_0 \text{ and } f(x_l^{(k)}) - v_k > \delta_k L_0 \tag{10}$$

have solutions for some $l = 1, \cdots, l_k$. Analytically, this requires δ_k to satisfy

$$A_k < \delta_k < B_k, \tag{11}$$

where

$$A_k = \frac{\min\{f(x_l^{(k)}) - v_k \,:\, l = 1, \cdots, l_k\}}{L_0}$$

and

$$B_k = \frac{\max\{f(x_l^{(k)}) - v_k \,:\, l = 1, \cdots, l_k\}}{L_0}.$$

When δ_k satisfies (11) and is small so to get close to A_k one has more bad sets to delete and less good sets to keep. In such case in the k-iteration step, the positions of the global minimizers are more accurately located, but as compensation, the algorithm has a greater computational cost. When δ_k is large so to get close to B_k, one has more good sets to keep and less bad sets to delete. In such case at the k-iteration step, the global minimizers are not so well located, but the computation cost is lower.

As analyzed in the last section, the design of the partition $\mathbb{N}_k$ is crucial. A design needs to balance the computation complexity, the algorithm effectiveness and its efficiency: At each sorting process, one wishes to delete a sufficient amount of bad sets so to quickly reduce the area of the set containing the minimizers. In practice, for simplicity, we divide the remaining set $C_k, (k > 1)$ into the same number of equal parts and take δ_k to be one proportional to m^{-k} for some integer m (see the parameter settings in the following section). In this situation, the diameter δ_k of the partition sets in the domain space and the interval $|I_k^{L_0}| = 3\delta_k L_0$ of the function values in the range space decrease exponentially.

An important issue to be addressed is related to the values L that are actually used in the algorithm. In the algorithm, the least upper bound L_0 of the bounds of the relative variations is crucial but maybe unknown in the practice. Determination of the least bound L_0, or even any bound L, is a similar problem. This is, in fact, a common problem for all Lipschitz-constant-dependent algorithms: If the Lipschitz constant is not known, one does not know what L should be used in the algorithm.

This L_0-paradox may be solved through a concept called *empirical-*, or *pseudo-L*, that we now formulate. We can use an arbitrary positive number $L > 0$, not necessarily a bound of the relative variations, to replace L_0 in the algorithm described by Theorem 1. We call such attempted L a *pseudo-L*. We will show that the algorithm with a pseudo-L, as substitution of L_0, may still be carried on and convergent. The corresponding algorithm is denoted as $\mathcal{A}_L$ that converges on the domain of f side to a set $\tilde{C}^L$, called the set of L-pseudo minimizers of f; and a single point v^L on the range of f side, called the L-pseudo minimum of f. The following theorem reveals the dynamic relation between the pseudo-L's and their corresponding algorithm outcomes.

Theorem 2. *(i) For any $L > 0$, the algorithm $\mathcal{A}_L$ converges to a compact set $\tilde{C}^L$ and a single value v^L such that $f(\tilde{C}^L) = \{v^L\}$; and, (ii) If $L_1 < L_2$, then $v^{L_1} \geq v^{L_2}$. Consequently, if $L \geq L_0$, then $\tilde{C}^L = \tilde{C}^{L_0} = \tilde{C}$, and $f(\tilde{C}^L) = \{v^{L_0}\} = \{v\}$.*

Proof of Theorem 2. (i). No matter how large or small L is, the algorithm procedure given in Theorem 1 converges on the domain side to a non-empty compact set $\tilde{C}^L$, and on the range side to a single point v^L. In fact, in the self-explanatory notation, due to the non-empty intersection property of the sequence of the compact sets $\{C_k^L\}_{k=1}^{\infty}$, there holds $\varnothing \neq \tilde{C}^L = \cap_{k=1}^{\infty} C_k^L$, and

$$\varnothing \neq f(\tilde{C}^L) = f(\cap_{k=1}^{\infty} C_k^L) \subset \cap_{k=1}^{\infty} f(C_k^L) \subset \cap_{k=1}^{\infty} I_k^L.$$

The set $f(\tilde{C}^L)$ can contain only one point, v^L, because

$$\lim_{k\to\infty} |I_k^L| = \lim_{k\to\infty} 3\delta_k L = 0.$$

(ii). First we note that the results of A_L is independent of the partitions in use. Assume $L_1 < L_2$. By using the same partitions at every k-step, with the pseudo-bounds L_1, L_2, respectively, we throw away the bad sets $C_l^{L_j,(k)}$ satisfying the condition

$$f(x_l^{(k)}) > v_k + \delta_k L_j, \qquad j = 1, 2.$$

Since for $j = 1$ the set $C_l^{L_j,(k)}$ is easier to satisfy the above inequality, or, in other words, easier to be bad, hence $C_{k+1}^{L_1}$ collects less sets than $C_{k+1}^{L_2}$. As a consequence, $v_{k+1}^{L_1} \geq v_{k+1}^{L_2}$. By taking limit $k \to \infty$, we have $v^{L_1} \geq v^{L_2}$. The proof is complete. □

On one hand, L_0 is difficult to find. On the other hand, L_0 is not necessarily to be found. In order to produce the correct global minimum of the objective function a pseudo-L does not necessarily reach L_0. It is easily observed that large relative variations may not present around the global, or even local, minimizers or maximizers.

To treat the paradox, the GrS algorithm is combined with a pseudo-L scheme involving a finite sequence of pseudo-L_i's, that is $L_1 < \cdots < L_n$. The starting L_1 and the ending L_n, however, are the art part of the method. L_1 in our examples may be estimated through the first partitions. With concrete problems although gradients are not directly used, their global bounds may be used to derive a practical bound of the relative variations. Such example include n-best kernel approximation in reproducing kernel Hilbert spaces (Saitoh and Sawano[30] and references therein where the open domain of the objective function can also be reduced to a compact domain set). The stopping L_n may be decided based on observation of non-improving of the optimizing result: when the values v^{L_n} become stable, one can decide to end the process. Actual use of pseudo-L's in concrete questions is empirical, crucial, and is the true art part of the algorithm that requires further and deep studies.

In practice, solving a global optimization problem is a trade-off problem: It is to find balance between accuracy and cost.[22] When employing the same set of consecutive partitions for a larger L, the convergence of $\mathcal{A}_L$ may be slower due to the fact that at each iteration step a less number of bad sets are thrown away. Larger pseudo-L's result in better accuracy but rise higher computational costs. Small pseudo-L's on the other hand may not give the right results but have fast convergence. The executable description of the GrS algorithm is shown in Algorithm 1.

Algorithm 1 GrS Algorithm

Require: $f(x) = f(x_1, x_2, \ldots, x_n)$; $D = \{(x_1, x_2 \ldots x_n) | x_i \in [a_i, b_i], i = 1, 2, \cdots, n\}$ is the domain.
Ensure: The L-pseudo minimizers, $\tilde{C}^L$; The L-pseudo minimum, v^L.
1: **Initialize:** Initial partition $\mathbb{N}_1$; Two thresholds for termination, η_1, η_2.
2: Calculate function values of all the center points, $x_l^{(1)}$;
3: Calculate the initial *pseudo*-L, $L = L_1$;
4: For $x \in C^{(1)}$, find $v_1 = \min f(x)$ and the corresponding $\widetilde{x} = \arg\min f(x)$;
5: Set $j = 1$;
6: **while** $err_1 > \eta_1$ & $err_2 > \eta_2$ **do**
7: Calculate δ_j based on the partition $\mathbb{N}_j$;
8: Obtain $x_l^{(j)}$ when $f(x_l^{(j)}) \leq v_j + \delta_j L$;
9: In $\mathbb{G}_j$, find $v_j = \min f(x)$ and the corresponding $\widetilde{x} = \arg\min f(x)$;
10: Set $err_1 = L\delta_j$, $err_2 = \delta_j$, $j :- j + 1$;
11: **end while**
12: Update L and execute Step 4-11;
13: Determine whether there are multiple solutions to obtain the final $\tilde{C}^L$ and v^L.
14: **return** $\tilde{C}^L$ and v^L.

5 | NUMERICAL EXPERIMENTS

5.1 | Benchmark functions

Typical test functions for general optimization problems are provided in the specialized reports by Gavana[31] and Jamil and Yang et al.[32] We selected the test functions commonly referenced in the above two reports to ensure that they have well-defined standards and properties. In total, there are 141 commonly referred test functions. For comparison with other optimization algorithms, we also tested the top 10 hardness optimization functions reported in Gavana,[31] seven of which are already included in the 141 aforementioned functions. We added three additional functions and resulted in a pool of 144 functions. Of the 144 functions, nine functions are discontinuous (including functions Corana, Csendes, Damavandi, Helical Valley, Keane, PenHolder, SchmitVetters, Step, and Tripod), and one has inconsistent numbers of variables and dimensions (function: Cola). Hence, these functions are excluded from our test function pool. We tested all the remaining 134 functions, which cover the characteristics of differentiability, separability, scalability, and modality.

Each of the 134 benchmark functions is defined by a formula in which some are dependent on a dimension parameter n. By letting n be concrete and different positive integers, we have a pool of more than 134 formulas, each of which we call a sample formula. The testing is performed on the sample formulas. As to the value of n, we set the following rule: if the GrS algorithm can find the optimization result in 4000 s, then we consider the sample formula as tested. In our experiments, the highest dimension parameter of all the tested sample formulas is 12. In total, 248 sample formulas were tested by automatic implementation under the same parameter settings.

In the experiments, we found that 61 out of 248 sample formulas (corresponding to 33 functions) provided by Gavana[31] and Jamil and Yang[32] had some problems. The problems can be classified into the following four categories.

(I) The minimum does not match the function value at the provided minimizer(s). This case can be verified by directly inputting the minimizer(s) into the function.
(II) The minimizers provided are incomplete. We found more minimizers and verified them too.
(III) Our algorithm found a strictly smaller minimum than the provided value.[31,32] The new minimum is also verified.
(IV) The given formula is incorrect. We found this problem by comparing Gavana[31] to Adorio and Diliman.[33]

Table 1 illustrates the 134 benchmark functions in the experiments and the problem types are labelled on the problematic functions. The functions for which our algorithm was unsuccessful are marked with * (for a function, as long as one of the dimensions is unsuccessful, the function is marked as unsuccessful).

5.2 | Experimental settings

The experiments were performed with MATLAB R2017b on a desktop computer with an Intel Core i9-10920X CPU at a 3.5 GHz clock frequency. In the experiments, for all the 2- and 3-dimensional functions, the side length along each dimension was split into 60 equal segments. For all the functions with more than three dimensions, the side length along each dimension was split into two equal segments. The stopping criteria on each L_i, $(i = 1, 2, \cdots, n)$ was set to $\delta_k L_i \leq 0.001$ or $\delta_k \leq 0.001$ holds for the first time.

TABLE 1 List of benchmark functions in alphabetical order including the problematic functions labelled with problem types

Index	Functions		
1-3	Ackley	Adjiman	*Alpine01
4-6	Alpine02 (I)	BartelsConn	Beale
7-9	Bird	Bohachevsky	BoxBetts
10-12	Branin01	Branin02	Brent
13-15	Brown	Bukin02 (III)	Bukin04
16-18	Bukin06	CarromTable	Chichinadze
19-21	Colville	CosineMixture (III)	CrossInTray
22-24	*CrossLegTable	Cube	Deb01 (III)
25-27	Deb02	DeckkersAarts	*DeVilliersGlasser01
28-30	*DeVilliersGlasser02	DixonPrice (I)	Dolan (III)
31-33	Easom	EggCrate	EggHolder
34-36	ElAttarVidyasagarDutta	Exp2 (I)	Exponential
37-39	FreudensteinRoth	Giunta	GoldsteinPrice
40-42	*Griewank	Gulf	Hansen (III)
43-45	Hartman3	Hartman6	*Himmelblau (I)
46-48	HolderTable (I)	Hosaki	JennrichSampson
49-51	Langermann	Leon	Matyas
52-54	McCormick	MieleCantrell	Mishra01 (II)
55-57	Mishra02	Mishra03 (III)	Mishra04 (I)
58-60	Mishra05	Mishra06 (III)	*Mishra07 (II)
61-63	Mishra08	Mishra09	Mishra10
64-66	Mishra11 (II)	*Parsopoulos	*Pathological
67-69	Paviani	Pinter	Powell
70-72	Price01	Price02	Price03 (I)
73-75	*Price04	Qing	Quadratic
76-78	*Quintic (II)	Rana (III)	Ripple01
79-81	Ripple25	Rosenbrock	RosenbrockModified (III)
82-84	RotatedEllipse01	RotatedEllipse02	Salomon
85-87	Sargan	Schaffer01	Schaffer02
88-90	*Schaffer03 (I)	Schaffer04 (III)	Schwefel01
91-93	Schwefel02	Schwefel04	Schwefel06
94-96	Schwefel20	Schwefel21	Schwefel22
97-99	Schwefel36	Shekel05 (III)(IV)	Shekel07 (III)(IV)
100-102	Shekel10 (III)(IV)	Shubert01	Shubert03 (III)
203-105	Shubert04 (I)(III)	SineEnvelope (I)(II)(III)	SixHumpCamel
106-108	Sphere	StyblinskiTang	TestTubeHolder (II)
109-111	ThreeHumpCamel	Treccani	Trefethen
112-114	Trid	*Trigonometric01	*Trigonometric02
115-117	Ursem01	Ursem03	Ursem04
118-120	UrsemWaves	VenterSobiezcczanskiSobieski	Watson (I)
121-123	Wavy	WayburnSeader01	*WayburnSeader02 (II)
124-126	Weierstrass (I)	Whitley	Wolfe
127-129	Xinsheyang01	XinSheYang02	XinSheYang03
130-132	XinSheYang04	Zacharov	Zettl
133-134	Zimmerman (IV)	Zirilli	

TABLE 2 Success rates of the GrS algorithm in finding the global minimizers

	Tested	Successful	Success rate (%)
Functions	134	119	88.81
Formulas	248	224	90.32

An initial L_1 is estimated with the maximum absolute value of the first-order difference quotients about all the center points at the first partition level. To find a suitable upper bound L, multiple runs of the GrS algorithm are executed on pseudo-upper bounds $L = L_1, L_2, \cdots, L_n$,where $L_i = 2^{i-1}L_1, (i = 2, 3, \cdots, n)$. In fact, based on Theorem 2, after a finite number of steps, L_{n-1} will become larger than L_0, and the minimum values between two consecutive GrS runs will no longer decrease; i.e., $v^{L_n} = v^{L_{n-1}}$. When v^{L_n} is observed to have such stability, the multiple runs can cease. In practice, to ensure the correctness of the results, when $v^{L_n} = v^{L_{n-1}}$ is observed, a few more GrS runs can be applied to ensure optimality.

TABLE 3 Test results for the top 10 hardness optimization functions listed in Gavana[31]

Hardness ranking	Benchmark function	Overall success rate against ground truth (%)	Result of GrS
1	DeVilliersGlasser02	0.00	Failure
2	Damavandi	0.25	Discontinuous
3	CrossLegTable	0.83	Failure
4	XinSheYang03	1.08	Success
5	SineEnvelope∗	2.17	Success
6	Whitley	4.92	Success
7	Zimmerman∗	4.92	Success
8	Griewank	6.08	Failure
9	Trefethen	6.58	Success
10	Bukin06	6.83	Success

Note: Test functions with ∗ indicate that there is a problem with the formula or its ground truth.

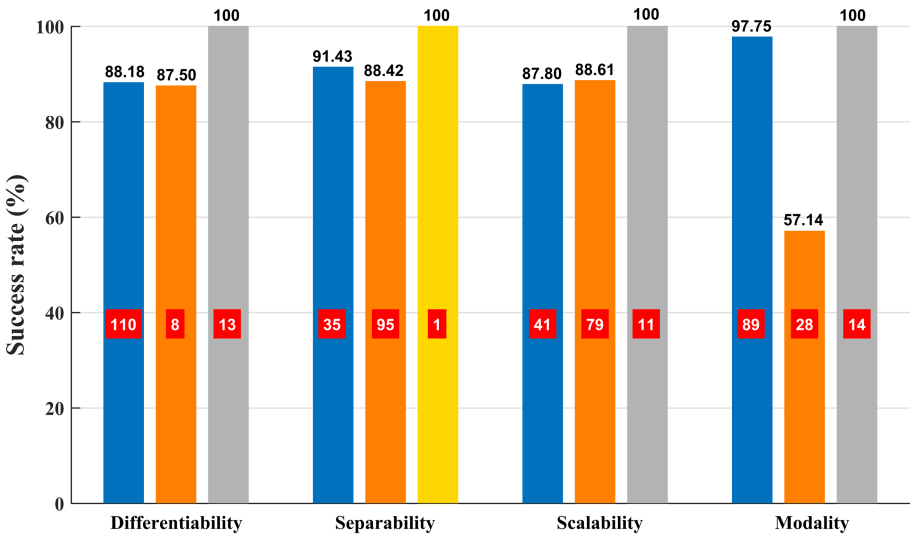


FIGURE 3 Statistical results on the characteristics of the test functions. The bars with different colors correspond to different types of characteristics. The blue bars represent differentiable, separable, scalable and unimodal types, respectively. The orange ones represent non-differentiable, non-separable, non-scalable and multimodal types. The gray ones are with unmentioned types. The yellow one is the partially-separable type. The white numbers with red background are the corresponding test function numbers [Colour figure can be viewed at wileyonlinelibrary.com]

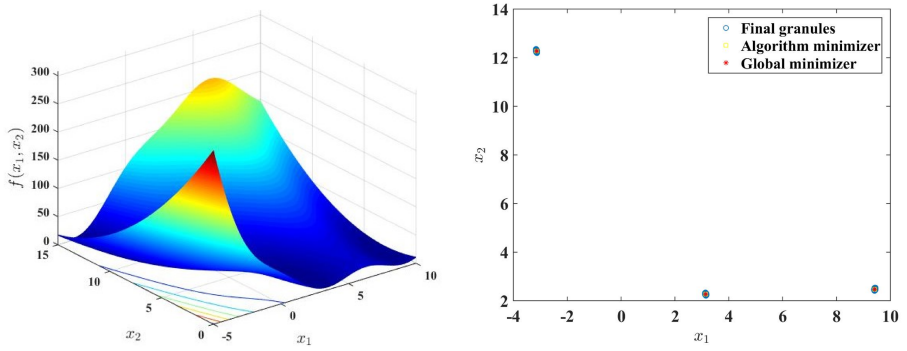
TABLE 4 Characteristics of the Jones test set

Index	Function	Dimension	Domain	Number of local minimizers	Number of global minimizers
1	Shekel 5	4	$[0,10]^4$	5	1
2	Shekel 7	4	$[0,10]^4$	7	1
3	Shekel 10	4	$[0,10]^4$	10	1
4	Hartman 3	3	$[0,1]^3$	4	1
5	Hartman 6	6	$[0,1]^6$	4	1
6	Branin RCOS	2	$[-5,10]\times[0,15]$	3	3
7	Goldstein and Price	2	$[-2,2]^2$	4	1
8	Six-Hump Camel	2	$[-5,5]^2$	6	2
9	2D Shubert	2	$[-10,10]^2$	760	18

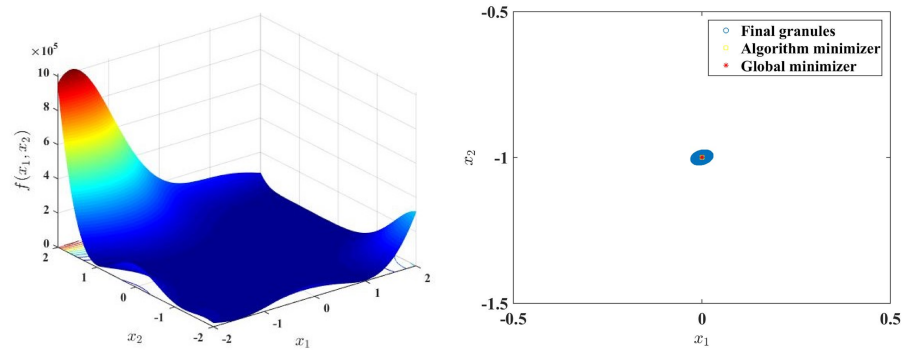
In our experiments, one more run is applied on $L_{n+1} = (2^{n-1}+1)L_1$. This is a trade-off process because the computational cost rapidly increases as the GrS iteration number increases. In summary, if $v^{L_{n+1}} = v^{L_n} = v^{L_{n-1}}$, then the multiple runs of the GrS algorithm finally stop. In this situation, GrS is ran at least three times in our experiments.

5.3 | Results and discussions on benchmark functions

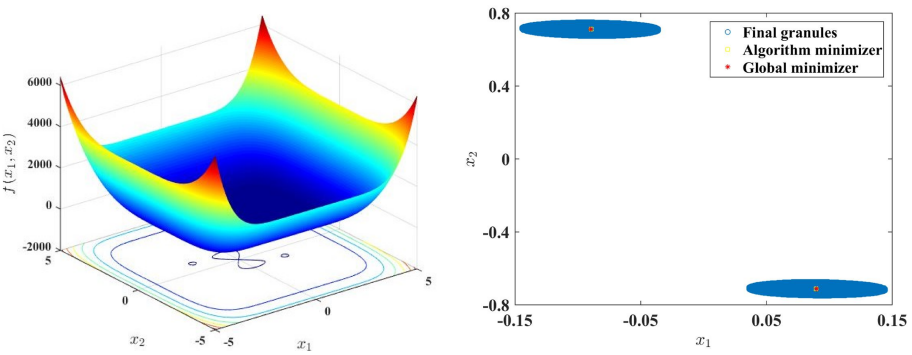
Among all the tested formulas, GrS correctly yielded both the minimum and minimizers for 224 sample formulas, corresponding to 119 functions. In the experimental results of 11 functions (19 sample formulas), the minimums yielded



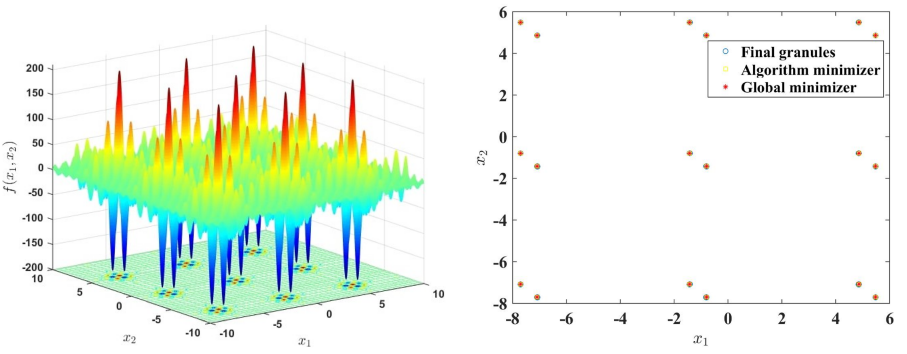
(A) Branin RCOS



(B) GoldsteinPrice



(C) Six-Hump Camel



(D) 2D Shubert

FIGURE 4 Visualization of four two-dimensional functions on the Jones test set. The original function graphs are shown on the left, while the graphs on the right display the final granules, algorithm minimizers and global minimizers [Colour figure can be viewed at wileyonlinelibrary.com]

by GrS approximate the ground truths, but the minimizers are incorrect. The analysis shows that this inconsistency is caused by an improperly set termination threshold or an underestimated pseudo-L value. By employing smaller termination thresholds or larger L values, or by allowing computing times greater than 4000 s, the problem of the inconsistency

can be solved. The experimental results of four other functions are different from both the reference provided minimum and minimizers. The analysis of the geometric nature of the function graphs shows that the failures are mostly caused by the first estimation of the pseudo-L in use. This suggests that a larger pseudo-L should be used to increase the effectiveness of the algorithm. With the improved L values, one can successfully find the correct solutions. The success rates of the benchmark functions and formulas are 88.81% and 90.32%, respectively, as listed in Table 2. If, like the other studies,[19,27,31] only finding the global minimum is regarded as a success, then our success rate on testing formulas is 98%.

To compare our algorithm with existing ones, in Table 3, we illustrate our test results on the top 10 benchmark optimization functions listed in the "Hardness" table of Gavana,[31] which provides algorithm comparison among 12 popular global optimization algorithms. The overall success rates in the table were reported to be obtained by running all the available global optimizers against all the test functions for a collection of 100 random starting points and then averaging the successful minimizations across all the optimizers.[31] As presented in Table 3, the hardest function for which our algorithm succeeds is XinSheYang03 (No. 4 in the top 10 "Hardness" list), and the overall success rate for XinSheYang03 is merely 1.08% for all the other 12 global optimizers mentioned in Gavana.[31] The non-differentiability of the function makes unavailability of the gradient-dependent methods. This is one of the reasons for the low success rate. This example shows that the gradient-free GrS is an effective algorithm regardless differentiability properties of the objective function. By adding the overall success rate of the top 10 hardest functions in Table 2, we can conclude that the overall success rate of the other 12 popular optimizers on the average of the 10 hardest problems is 3.366%, while the overall success rate of GrS on the average of the 10 hardest problems is 60%.

Apart from the continuity and finite-L_0 property, the GrS algorithm does not require any other properties from the objective functions. Our benchmark functions include non-differentiable, non-separable, scalable and multimodal types, for which global minimizers are generally difficult to obtain.[32] The detailed statistical results on the success rate are presented in Figure 3. In total, 131 functions that are commonly referred by Jamil and Yang[32] and Gavana[31] are listed. The characteristics of the exceptional three functions from Table 3 are not labeled so that they are not counted.

Pseudo-L is an important parameter used and estimated along with the algorithm process. In 201 out of 224 successfully tested formulas, L is estimated thrice. Among all the tested sample formulas, the maximal number of estimations for L is 11. The detailed experiment results on all the sample formulas can be found in the webpage of the Experiment Results[†].

5.4 | Results and discussions on the Jones test set

We select nine classical functions, that is the commonly used Jones test set, from the 134 benchmark functions to demonstrate the performance of GrS against benchmark functions. The significant characteristics of the Jones test set are illustrated in Table 4. All the nine functions are multimodal with at least three local minimas. Among them, 2D Shubert has the most global minimizers which number is 18.

The GrS algorithm showed excellent performance for the nine functions and successfully found all the global minimizers. Figure 4 shows the visualization results of the last four two-dimensional functions in Table 4. The left graphs displays the morphological characteristics of four benchmark functions. The right graphs compare the position difference between our found minimizers and the global minimizers, which are marked with red stars and yellow squares on the two-dimensional coordinate planes, respectively. The blue circles are the final remaining granules when the algorithm terminates. The visualization results indicate the feasibility of the GrS algorithm when facing a single global minimizer or multiple global minimizers.

Table 5 exhibits the final numerical results calculated by the GrS algorithm on the Jones test set. The global minima are referred from,[34] and the global minimizers are referred from Jamil and Yang[32] and Gavana.[31] The smaller value between the global minimum and the algorithm minimum is in bold. In addition to the four two-dimensional functions, our algorithm also shows satisfactory performance on the other functions. Moreover, it is noticed that the GrS algorithm found a smaller minimum than the given global minimum for Hartman 3.

[†]https://www.fst.um.edu.mo/personal/lmzhang/experiment_results.

TABLE 5 Numerical results on the Jones test set

Index	Global minimum	Algorithm minimum	Global minimizer	Algorithm minimizer
1	−10.15320	−10.15320	[4,4,4,4]	[4.00009,4.00009, 4.00009,4.00009]
2	**−10.40294**	**−10.40294**	[4,4,4,4]	[4.00070,4.00070, 3.99948,3.99948]
3	**−10.53641**	**−10.53641**	[4,4,4,4]	[4.00070,4.00070, 3.99979,3.99948]
4	−3.86278	**−3.86280**	[0.1, 0.55592, 0.85218]	[0.11484, 0.55547, 0.85234]
5	**−3.32237**	−3.30153	[0.20169, 0.15001, 0.47687, 0.27533, 0.31165, 0.65730]	[0.20313, 0.14063, 0.48438, 0.26563, 0.29688, 0.67188]
6	**0.39789**	**0.39789**	[$-\pi$,12.275], [π,2.275], [9.42478,2.475]	[9.42456,2.47485], [3.12476,2.23999], [−3.12573,12.18628]
7	**3**	3.00003	[0,-1]	[0.00026,−0.99974]
8	**−1.03163**	**−1.03163**	[0.08984,−0.71266], [−0.08984,0.71266]	[0.08952,−0.71257], [−0.08952,0.71257]
9	**−186.73091**	−186.73066	[−7.08351,4.85806] (and many others)	[−7.08366,4.85775] (and many others)

6 | CONCLUSIONS

The proposed GrS method is based on intuitive and simple mathematical formulation and has an easily implementable algorithm. The algorithm is effective and efficient for any finite dimension, offering the complete set of solutions within the prescribed accuracy. If the Lipschitz constant L_0 is known, then the algorithm has theoretically no limitation. If the Lipschitz constant is unknown, GrS is incorporated with a pseudo-Lipschitz process to solve the black-box Lipschitzian problem. However, it practically requires a large but relative moderate computer storage capacity. Significant topics for future studies would include finding effective pseudo-L's and deeper relations between the used L and GrS.

ACKNOWLEDGEMENTS

This work was funded by the Science and Technology Development Fund of Macau SAR 0123/2018/A3, 0060/2021/A and Multi-year Research Grant MYRG2018-00111-FST.

CONFLICT OF INTERESTS

This work does not have any conflict of interests.

ORCID

Tao Qian https://orcid.org/0000-0002-8780-9958
Liming Zhang https://orcid.org/0000-0002-2664-8193

REFERENCES

1. Qian T, Wegert E. Optimal approximation by Blaschke forms. *Complex Var Elliptic Equ*. 2013;58:123-133.
2. Qian T. Cyclic AFD algorithm for best rational approximation. *Math Methods Appl Sci*. 2014;37(6):846-859.
3. Qian T, Wang J, Lai W. An enhancement algorithm for cyclic adaptive Fourier decomposition. *Appl Comput Harmon Anal*. 2019;47(2):516-525.
4. Bargiela A, Pedrycz W. Granular computing. *Handbook on Computational Intelligence: Fuzzy Logic, Systems, Artificial Neural Networks, and Learning Systems*. World Scientific: 2016.
5. Lin TY. Granular computing: practices, theories, and future direction. *Encycl Complexity Syst Sci*. 2009;2009:4339-4355.
6. Pedrycz W, Skowron A, Kreinovich V, eds. *Handbook of Granular Computing*: John Wiley & Sons; 2008.
7. Lera D, Sergeyev YD. GOSH: derivative-free global optimization using multi-dimensional space-filling curves. *J Glob Optim*. 2018;71(1):193-211.
8. Paulavičius R, Žilinskas J. *Simplicial Global Optimization*. New York: Springer; 2014.
9. Sergeyev YD, Kvasov DE. *Deterministic Global Optimization: An Introduction to the Diagonal Approach*. New York: Springer; 2017.
10. Horst R, Tuy H. *Global Optimization: Deterministic Approaches*. Springer Science and Business Media; 2013.
11. Horst R, Pardalos PL, eds.. *Handbook of Global Optimization*. Springer Science and Business Media; 2013.
12. Lera D, Sergeyev YD. Acceleration of univariate global optimization algorithms working with Lipschitz functions and Lipschitz first derivatives. *SIAM J Optim*. 2013;23(1):508-529.
13. Sergeyev YD, Kvasov DE. Global search based on efficient diagonal partitions and a set of Lipschitz constants. *SIAM J Optim*. 2006;16:910-937.
14. Strongin RG, Sergeyev YD. *Global Optimization with Non-Convex Constraints: Sequential and Parallel Algorithms*: Springer Science and Business Media; 2013.
15. Jones DR, Perttunen CD, Stuckman BE. Lipschitzian optimization without the Lipschitz constant. *J Optim Theory Appl*. 1993;79:157-181.
16. Wang YB, Qian T. Pseudohyperbolic distance and n-best rational approximation in H^2 space. *Math Methods Appl Sci*. 2021;44(11):8497-8504.
17. Floudas CA, Akrotirianakis IG, Caratzoulas S, Leyer CA, Kallrath J. Global optimization in the 21st century. Advances and challenges. *Comput Chem Eng*. 2005;29:1185-1202.
18. Pintér JD, Optimization. *Global Scientific and Engineering Case Studies*: Springer Science and Business Media; 2006.
19. Törn A, Žilinskas A, Global optimization, eds. Springer; 1989.
20. Floudas CA, Gounaris CE. A review of recent advances in global optimization. *J Glob Optim*. 2009;45(1):3-38.
21. Horst R, Pardalos PL, Van Thoai N. *Introduction to Global Optimization*. Springer Science and Business Media; 2000.
22. Barhen J, Protopopescu V, Reister D. TRUST: A deterministic algorithm for global optimization. *Science*. 1997;276:1094-1097.
23. Booker AJ, Dennis JE, Frank PD, Serafini DB, Torczon V, Trosset MW. A rigorous framework for optimization of expensive functions by surrogates. *Struct Optim*. 1999;17:1-13.
24. Huyer W, Neumaier A. SNOBFIT-stable noisy optimization by branch and fit. *ACM Trans Math Softw*. 2008;35(2):1-25.

25. Jones DR. A taxonomy of global optimization methods based on response surfaces. *J Glob Optim*. 2001;21(4):345-383.
26. Rios LM, Sahinidis NV. Derivative-free optimization: a review of algorithms and comparison of software implementations. *J Glob Optim*. 2013;56(3):1247-1293.
27. Liberti L, Laculan N, eds. *Global Optimization: From Theory to Implementation*. Springer Science and Business Media; 2006.
28. Pintér JD. *Global Optimization in Action-Continuous and Lipschitz Optimization. Algorithms, Implementations and Applications*: Springer Science and Business Media; 2013.
29. Hewitt E, Stromberg K. *Real and Abstract Analysis: A Modern Treatment of the Theory of Functions of a Real Variable*. Springer-Verlag; 2013.
30. Saitoh S, Sawano Y. *Theory of Reproducing Kernels and Applications*. Singapore: Springer; 2016.
31. Gavana A. Test function index. Accessed March 4, 2021. http://infinity77.net/global_optimization/test_functions.html#test-functions-index
32. Jamil L, Yang XS. A literature survey of benchmark functions for global optimisation problems. *Int J Math Model Numer Optim*. 2013;4(2):150-194.
33. Adorio EP, Diliman UP. MVF-multivariate test functions library in C for unconstrained global optimization. 2005. http://www.geocities.ws/eadorio/mvf.pdf
34. Liu QF, Cheng WY. A modified DIRECT algorithm with bilevel partition. *J Glob Optim*. 2014;60(3):489-499.

How to cite this article: Qian T, Dai L, Zhang L, Chen Z. Granular sieving algorithm for selecting best *n* parameters. *Math Meth Appl Sci*. 2022;45(12):7495-7509. doi:10.1002/mma.8254